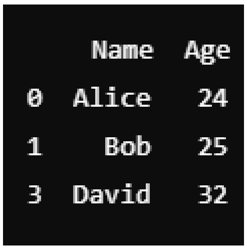
16. Removing duplicate values

Pandas provide `drop_duplicate()` function to remove duplicate values from a DataFrame. This function returns a DataFrame with duplicate rows removed.
You can customize the behavior of `drop_duplicates()` to fit your specific needs, such as considering certain columns for identifying duplicates or keeping the first or last occurrence of duplicates.

Ex:

1. Remove duplicates

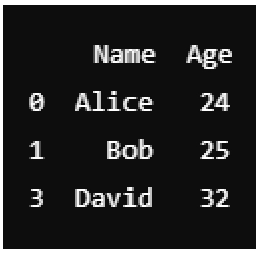
```
import pandas as pd

data = {
        'Name': ['Alice', 'Bob', 'Alice', 'David', 'Bob'],
    'Age': [24, 25, 24, 32, 25]}
df = pd.DataFrame(data)

df_no_duplicates = df.drop_duplicates()
print(df_no_duplicates)
```



```
    Name  Age
0  Alice   24
1    Bob   25
3  David   32
```

2. Remove duplicates on specific columns
   If you want to remove duplicates based on certain columns, you can specify those columns using the `subset` parameter.

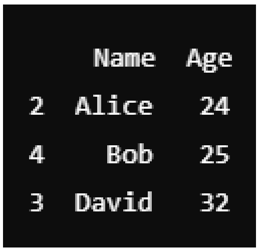
```
df_no_duplicates_name = df.drop_duplicates(subset=['Name'])
print(df_no_duplicates_name)
```



```
    Name  Age
0  Alice   24
1    Bob   25
3  David   32
```

3. By default, `drop_duplicates()` keeps the first occurrence of each duplicate row. You can change this behavior to keep the last occurrence using the `keep` parameter.

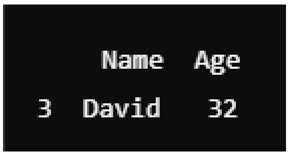
```
df_no_duplicates_last = df.drop_duplicates(keep='last')
print(df_no_duplicates_last)
```



```
    Name  Age
2  Alice   24
4    Bob   25
3  David   32
```

4. Remove all duplicates
   set `keep=False`

```
df_unique = df.drop_duplicates(keep=False)
print(df_unique)
```



```
    Name  Age
3  David   32
```

5. In-place removal
   Remove duplicates in place, without creating a new DataFrame.

```
df.drop_duplicates(inplace=True)
print(df)
```